developer-be

说明，代码大部分基于v1.2版本分析

核心模块

简称	功能	描述
tool	开发工具管理	提供给开发者使用的App开发工具，开发者可以在这里下载使用工具，同时可以通过将自研的工具通过上传分享给社区开发者使用。
api	API管理	分为MEPAPI，MEP-ECOAPI，开发者平台API。包含详细的API描述文档，供开发者查看。
test	APP测试	开发者平台提供共用的测试环境，开发人员完成App开发后，可以在平台上创建测试任务对App功能进行测试。
workspace	工作空间	工作空间是给开发者提供的项目管理平台，开发者可以通过创建项目，完成App的开发-测试-发布的整个流程。

整体流程

http://docs.edgegallery.org/zh_CN/release-v1.0/Projects/Developer/Developer_Features.html

插件管理

插件管理个人理解主要用于营造生态，开发者将能力涉及的SDK封装为IDE插件上传至平台，平台提供上传下载列表查看等。接口有：

技术上只是一些db的CRUD，并无特别。

能力中心

同样用于营造生态，这里是eg提供的能力入口，包括**分组**与**分组下的能力**。

其中，能力分组接口只是简单的db操作，不再赘述。在某分组下，创建能力，能力的数据结构定义为：

```
public class OpenMepCapabilityDetail {
 private String detailId;

 private String groupId;
 private String service;
 private String serviceEn;
 private String version;
 private String description;
 private String descriptionEn;

```

```
11  private String provider;
12  // download or show api
13  // 记录能力的api和说明文档信息
14  private String apiFileId;
15  private String guideFileId;
16  private String guideFileIdEn;
17  private String uploadTime;
18  //存疑
19  private int port;
20  private String host;
21  private String protocol;
22  //
23  private String appId;
24  private String packageId;
25  private String userId;
```

工作空间

工作空间是创建/部署app的关键，主要分为

创建项目

在工作空间创建project，并通过选择能力中心的mep能力，进行mep依赖定义，具体接口包括了：

- 3.16 POST open project api
- 3.17 GET project atp task

其中project的定义为：

```
public class ApplicationProject {
 //...字段约束正则，省略
 // normal data start
 private String id;
 //项目类型 new/集成
 private EnumProjectType projectType;

 @Pattern(regexp = "^(?!_)(?!-)(?!\\s)(?!.*?_$)(?!.*?-$)(?!.*?\\s$)[a-zA-Z0-9_-]{4,32}$", message = NAME_MSG)
 private String name;
  @Pattern(regexp = "^[\\w\\-][\\w\\-\\s.]{0,9}$", message = VERSION_MSG)
  private String version;
  @Pattern(regexp = "^\\S.{0,29}$", message = PROVIDER_MSG)
  private String provider;

  private List<String> platform; //架构，x86 arm32 arm64
  private EnumDeployPlatform deployPlatform; //部署平台 k8s / 虚机

  // add to match app store
  private String type; //项目类型，如视频类
  private List<String> industry; //场景，如智慧园区
  @Pattern(regexp = "^(?!\\s)[\\S.\\s\\n\\r]{1,128}$", message = DESCRIPTION_MSG)
  private String description;
  private String iconFileId; //图标文件

  private EnumProjectStatus status;//项目状态，online deploying deployed deployFailed testing tested released
  private List<OpenMepCapabilityGroup> capabilityList;//重要，集成的能力列表

  private String lastTestId;
  private String userId;
  private String createDate;
  private String openCapabilityId; //首次创建为null，发布后更新
```

以一个实际的创建project为例，操作上集成了**服务发现**和**AI图像识别**能力，其json为：

```
{
 "id": "f81b491d-d011-4027-9628-de7739d0747f",
```

```
  "projectType": "CREATE_NEW",
  "name": "delete",
  "version": "v1.0",
  "provider": "Huawei",
  "platform": [
  "X86"
  ],
   "deployPlatform": "KUBERNETES",
   "type": "Video Application",
   "industry": [
   "Smart Park"
   ],
   "description": "test",
   "iconFileId": "c165a26a-7bb1-40fe-844b-8396b4885787",
   "status": "ONLINE",
   "capabilityList": [
   {
   "groupId": "c0db376b-ae50-48fc-b9f7-58a609e3ee12",
   "oneLevelName": "平台基础服务",
   "oneLevelNameEn": "Platform services",
   "twoLevelName": "服务治理",
   "twoLevelNameEn": "Service governance",
   "type": "OPENMEP",
   "description": "EdgeGallery平台为APP提供服务注册、发现、订阅等相关功能。",
   "descriptionEn": "The EdgeGallery platform provides APP with related fu
nctions such as service registration, discovery, and subscription.",
   "iconFileId": "35a52055-42b5-4b5f-bc2b-8a02259f2572",
   "author": "admin",
   "selectCount": 2,
   "uploadTime": "Jun 14, 2021 6:00:00 PM",
   "capabilityDetailList": [
   {
   "detailId": "143e8608-7304-4932-9d99-4bd6b115dac8",
   "groupId": "c0db376b-ae50-48fc-b9f7-58a609e3ee12",
   "service": "服务发现",
   "serviceEn": "service discovery",
   "version": "v1",
   "description": "EdgeGallery平台为APP提供服务注册、发现、订阅等相关功能。",
   "descriptionEn": "The EdgeGallery platform provides APP with related fu
nctions such as service registration, discovery, and subscription.",
```

```
"provider": "Huawei",
"apiFileId": "540e0817-f6ea-42e5-8c5b-cb2daf9925a3",
"guideFileId": "9bb4a85f-e985-47e1-99a4-20c03a486864",
"guideFileIdEn": "9ace2dfc-6548-4511-96f3-2f622736e18a",
"uploadTime": "2021-06-14 18:00:00.384+08",
"port": 8684,
"host": "service-discovery",
"protocol": "http",
"userId": "admin"
}
]
},
{
"groupId": "c0db376b-ae50-48fc-b9f7-58a609e3ee13",
"oneLevelName": "昇腾AI能力",
"oneLevelNameEn": "Ascend AI",
"twoLevelName": "AI图像修复",
"twoLevelNameEn": "AI Image Repair",
"type": "OPENMEP",
"description": "AI图像修复技术，可以快速帮助你去除照片中的瑕疵，你的照片你做主，一切问题AI帮你搞定。",
"descriptionEn": "AI image repair technology can quickly help you remove the blemishes in your photos. Your photos are up to you, and AI will help you solve all problems.",
"iconFileId": "56302719-8c85-4226-b01e-93535cdb2e42",
"author": "admin",
"selectCount": 0,
"uploadTime": "Jun 14, 2021 5:54:00 PM",
"capabilityDetailList": [
{
"detailId": "143e8608-7304-4932-9d99-4bd6b115dac9",
"groupId": "c0db376b-ae50-48fc-b9f7-58a609e3ee13",
"service": "AI图像修复",
"serviceEn": "AI Image Repair",
"version": "v1",
"description": "AI图像修复技术，可以快速帮助你去除照片中的瑕疵，你的照片你做主，一切问题AI帮你搞定。",
"descriptionEn": "AI image repair technology can quickly help you remove the blemishes in your photos. Your photos are up to you, and AI will help you solve all problems.",
"provider": "Huawei",
"apiFileId": "9ace2dfc-6548-4511-96f3-1f622736e182",
```

```
  "guideFileId": "9ace2dfc-6548-4511-96f3-2f622736e181",
  "guideFileIdEn": "9ace2dfc-6548-4511-96f3-2f622736e181",
  "uploadTime": "2021-06-14 17:54:00.384+08",
  "port": 0,
  "host": "",
  "protocol": "http",
  "userId": "admin"
  }
  ]
  }
  ],
  "lastTestId": null,
  "userId": "39937079-99fe-4cd8-881f-04ca8c4fe09d",
  "createDate": "2021-08-06 23:19",
  "openCapabilityId": null
}
```

后端实现上除了有对图标的文件操作为，基本都是db操作，不再赘述。

能力详情&应用开发

能力详情主要用于展示project依赖的mep平台能力，实现上接口复用了能力中心，不再赘述。应用开发几乎只是页面展示。

部署调测

部署调测中，主要涉及三步，分别为：

1. 上传app镜像
2. 配置部署文件
3. 部署调测

1. 上传镜像

镜像操作涉及如下接口：

3.13 POST add image to project

3.14 DELETE image of project

3.15 GET image of project

实现上，eg在大文件的上传上分2步进行，分别为**分块上传**和**merge**操作：

```
//upload
@ApiOperation(value = "upload image", response = ResponseEntity.class)
@ApiResponses(value = {
```

```
4   @ApiResponse(code = 200, message = "OK", response = ResponseEntity.class),
5   @ApiResponse(code = 400, message = "Bad Request", response = ErrorRespDto.class)
6   })
7   @RequestMapping(value = "/upload", method = RequestMethod.POST)
8   @PreAuthorize("hasRole('DEVELOPER_TENANT') || hasRole('DEVELOPER_ADMIN')")
9   public ResponseEntity uploadImage(HttpServletRequest request, Chunk chunk) throws IOException {
10   boolean isMultipart = ServletFileUpload.isMultipartContent(request);
11   if (isMultipart) {
12   MultipartFile file = chunk.getFile();
13   ...
14   File uploadDirTmp = new File(filePathTemp);
15   ...
16   Integer chunkNumber = chunk.getChunkNumber();
17   ..
18   //将一个个chunk在tmp下保存
19   File outFile = new File(filePathTemp + File.separator + chunk.getIdentifier(), chunkNumber + ".part");
20   InputStream inputStream = file.getInputStream();
21   FileUtils.copyInputStreamToFile(inputStream, outFile);
22   }
23   return ResponseEntity.ok().build();
24   }
25   //merge
26   @ApiOperation(value = "merge image", response = ResponseEntity.class)
27   @ApiResponses(value = {
28   @ApiResponse(code = 200, message = "OK", response = ResponseEntity.class),
29   @ApiResponse(code = 400, message = "Bad Request", response = ErrorRespDto.class)
30   })
31   @RequestMapping(value = "/merge", method = RequestMethod.GET)
32   @PreAuthorize("hasRole('DEVELOPER_TENANT') || hasRole('DEVELOPER_ADMIN')")
33   public ResponseEntity mergeImage(@RequestParam(value = "fileName", required = false) String fileName,
34   @RequestParam(value = "guid", required = false) String guid) throws IOException {
35   File uploadDir = new File(filePath);
36   ...
```

```
File file = new File(filePathTemp + File.separator + guid);
if (file.isDirectory()) {
//merge file
File[] files = file.listFiles();
if (files != null && files.length > 0) {
File partFile = new File(filePath + File.separator + fileName);
for (int i = 1; i <= files.length; i++) {
File s = new File(filePathTemp + File.separator + guid, i + ".part");
FileOutputStream destTempfos = new FileOutputStream(partFile, true);
FileUtils.copyFile(s, destTempfos);
destTempfos.close();
}
FileUtils.deleteDirectory(file);

//push image to repo
if (!pushImageToRepo(partFile)) {
return ResponseEntity.badRequest().build();
}
//delete all file in "filePath"
File uploadPath = new File(filePath);
FileUtils.cleanDirectory(uploadPath);

}
}
return ResponseEntity.ok().build();
}
```

在将镜像推送到repo的动作上，思路和sigma类似，通过docker client的api，解析manifest.json，调用docker push

```
private boolean pushImageToRepo(File imageFile) throws IOException {
 DockerClient dockerClient = getDockerClient(devRepoEndpoint, devRepoUsername, devRepoPassword);
 try (InputStream inputStream = new FileInputStream(imageFile)) {
 //import image pkg,执行 docker load -o {file}操作
 dockerClient.loadImageCmd(inputStream).exec();
 }
 ...
 //Unzip the image package, Find outmanifest.jsonmiddleRepoTags
 //解析manifest.json，得到image的tag和id
 File file = new File(filePath);
 boolean res = deCompress(imageFile.getCanonicalPath(), file);
```

```
12  String repoTags = "";
13  if (res) {
14  //Readmanifest.jsonContent
15  File manFile = new File(filePath + File.separator + "manifest.json");
16  String fileContent = FileUtils.readFileToString(manFile, "UTF-8");
17  String[] st = fileContent.split(",");
18  for (String repoTag : st) {
19  if (repoTag.contains("RepoTags")) {
20  String[] repo = repoTag.split(":\\[");
21  repoTags = repo[1].substring(1, repo[1].length() - 2);
22  }
23  }
24  }
25  LOGGER.debug("repoTags: {} ", repoTags);
26  String[] names = repoTags.split(":");
27  //Judge the compressed packagemanifest.jsoninRepoTagsAnd the value oflo
adAre the incoming mirror images equal
28  LOGGER.debug(names[0]);
29  List<Image> lists = dockerClient.listImagesCmd().withImageNameFilter(na
mes[0]).exec();
30  LOGGER.debug("lists is empty ?{},lists size {},number 0 {}", Collection
Utils.isEmpty(lists), lists.size(),
31  lists.get(0));
32  String imageId = "";
33  if (!CollectionUtils.isEmpty(lists) && !StringUtils.isEmpty(repoTags))
{
34  for (Image image : lists) {
35  LOGGER.debug(image.getRepoTags()[0]);
36  String[] images = image.getRepoTags();
37  if (images[0].equals(repoTags)) {
38  imageId = image.getId();
39  LOGGER.debug(imageId);
40  }
41  }
42  }
43  LOGGER.debug("imageID: {} ", imageId);
44  //拼装image name，需要结合eg部署的developer的harbor地址
45  String uploadImgName = new StringBuilder(devRepoEndpoint).append("/").a
ppend(devRepoProject).append("/")
46  .append(names[0]).toString();
47  //Mirror tagging, Repush
```

```
48  String[] repos = repoTags.split(":");
49  if (repos.length > 1 && !imageId.equals("")) {
50  //tag image，执行docker tag 重新打tag
51  dockerClient.tagImageCmd(imageId, uploadImgName,
repos[1]).withForce().exec();
52  LOGGER.debug("Upload tagged docker image: {}", uploadImgName);
53  //push image
54  try {
55  dockerClient.pushImageCmd(uploadImgName).exec(new PushImageResultCallba
ck()).awaitCompletion();
56  } catch (InterruptedException e) {...}
57  }
58  ...
59  return true;
60  }
```

2. 配置部署文件

部署配置文件用于定义在k8s环境部署时的yaml定义，可以通过**页面(1.2版本提供)或者上传文件**的方式定义。其中，文件操作的接口为：

- 6. File
  - 6.1 GET one file
  - 6.2 POST upload one file
  - 6.3 POST upload helm yaml
  - 6.4 GET helm yaml
  - 6.5 DELETE helm yaml
  - 6.6 POST get sample code
  - 6.7 GET one file return object
  - 6.8 GET sdk code
  - 6.9 GET file content
  - 6.10 POST pkg structure

```
1 public Either<FormatRespDto, HelmTemplateYamlRespDto> uploadHelmTemplateY
aml(MultipartFile helmTemplateYaml,
2  String userId, String projectId, String configType) throws IOException {
3  String content;
4  File tempFile;
5  //解析file
6  try {
7  tempFile = File.createTempFile(UUID.randomUUID().toString(), null);
8  helmTemplateYaml.transferTo(tempFile);
9  content = FileUtils.readFileToString(tempFile, Consts.FILE_ENCODING);
```

```
10  } catch (IOException e) {
11  ...
12  }
13  HelmTemplateYamlRespDto helmTemplateYamlRespDto = new HelmTemplateYamlRespDto();
14  String oriName = helmTemplateYaml.getOriginalFilename();
15  //err check ..
16  String originalContent = content;
17  content = content.replaceAll(REPLACE_PATTERN.toString(), "");
18  // verify yaml scheme
19  String[] multiContent = content.split("---");
20  List<Map<String, Object>> mapList = new ArrayList<>();
21  try {
22  for (String str : multiContent) {
23  Yaml yaml = new Yaml();
24  //靠yaml包做基本的格式校验
25  Map<String, Object> loaded = yaml.load(str);
26  mapList.add(loaded);
27  }
28  helmTemplateYamlRespDto.setFormatSuccess(true);
29  } catch (Exception e) {
30  ...
31  }
32  List<String> requiredItems = Lists.newArrayList("image", "service", "mep-agent");
33  // 验证，verify service,image,mep-agent
34  verifyHelmTemplate(mapList, requiredItems, helmTemplateYamlRespDto);
35  //...generate resp
36  }
37  return getSuccessResult(helmTemplateYaml, userId, projectId, originalContent, helmTemplateYamlRespDto,
38  configType, tempFile);
39
40  }
```

相应的，通过页面配置的接口为：

配置后的yaml文件大致内容为（**可以看到，除了业务应用busybox，yaml中描述了一个mep-agent，这个是重点，在mep-agent中单独分析**）：

```
---
apiVersion: v1
kind: Pod
metadata:
 name: busybox-pod
 namespace: default
 labels:
 app: busybox-pod
spec:
  containers:
  -
  name: busybox
  image: '{{.Values.imagelocation.domainname}}/{{.Values.imagelocation.project}}/busybox:lates'
  imagePullPolicy: Always
  env:
  -
  name: ""
  value: ""
  ports:
  -
  containerPort: 80
  command: '[\"top\"]'
  resources:
  limits:
  memory: 100Mi
  cpu: 1
  requests:
  memory: 100Mi
  cpu: 1
  -
  name: mep-agent
  image: '{{.Values.imagelocation.domainname}}/{{.Values.imagelocation.project}}/mep-agent:latest'
  imagePullPolicy: Always
  env:
  -
  name: ENABLE_WAIT
```

```
value: '"true"'
-
name: MEP_IP
value: '"mep-api-gw.mep"'
-
name: MEP_APIGW_PORT
value: '"8443"'
-
name: CA_CERT_DOMAIN_NAME
value: '"edgegallery"'
-
name: CA_CERT
value: /usr/mep/ssl/ca.crt
-
name: AK
valueFrom:
secretKeyRef:
name: '{{ .Values.appconfig.aksk.secretname }}'
key: accesskey
-
name: SK
valueFrom:
secretKeyRef:
name: '{{ .Values.appconfig.aksk.secretname }}'
key: secretkey
-
name: APPINSTID
valueFrom:
secretKeyRef:
name: '{{ .Values.appconfig.aksk.secretname }}'
key: appInsId
volumeMounts:
-
name: mep-agent-service-config-volume
mountPath: /usr/mep/conf/app_instance_info.yaml
subPath: app_instance_info.yaml
volumes:
-
name: mep-agent-service-config-volume
configMap:
```

```
 name: '{{ .Values.global.mepagent.configmapname }}'
---
apiVersion: v1
kind: Service
metadata:
 name: busybox-svc
 namespace: default
 labels:
 svc: busybox-svc
spec:
 ports:
 -
 port: 80
 targetPort: 80
 protocol: TCP
 nodePort: 32115
 selector:
 app: busybox-svc
 type: NodePort

```

具体实现不再赘述。这里需要注意，当完成部署文件配置后，或上传yaml成功后，除了调用post yaml类接口，**此处还调用了test-config的PUT接口，test-config数据主要用于记录在整个部署过程中，project的状态变化以及新增/更改的重要信息。比如在完成文件配置后的testConfig PUT 接口发送了如下数据：**


```
{
 "testId": "8dc00669-34b1-4fe9-b64e-092b30463f5a",
 "projectId": "4c75d3a7-b82e-4373-92f5-e03b1fedd5a2",
 "platform": "KUBERNETES",
 "deployFileId": "f51e2f65-ee30-4757-b80c-15d83f0a41c7",
 "privateHost": false,
 "pods": null,
 "deployStatus": "NOTDEPLOY",
 "stageStatus": null,
 "hosts": null,
 "errorLog": null,
 "workLoadId": null,
 "appInstanceId": "0104eec8-7803-4caa-a01a-a8d85d6ee778",
 "deployDate": null,
 "lcmToken": null,
```

```
  "agentConfig": null,
  "imageFileIds": null,
  "appImages": null,
  "otherImages": null,
  "appApiFileId": null,
  "accessUrl": null,
  "packageId": null,
  "nextStage": "csar"
}
```

其test-config接口的操作如下：

3.8 POST create test config

3.9 PUT modify test config

3.10 GET one test-config

3. 部署调测：

部署调测的操作接口为3.6 POST deploy one project

在实现上，从大的方向上走，有两个方面

- **DB操作：**

```
public Either<FormatRespDto, ApplicationProject> deployProject(String userId, String projectId, String token) {
  // 因为在上传yaml那里创建了testConfig，此处获取
  List<ProjectTestConfig> testConfigList = projectMapper.getTestConfigByProjectId(projectId);
  ...
  // only one test-config for each project
  ProjectTestConfig testConfig = testConfigList.get(0);
  // check status
  ...
  // update test-config status
   //创建appInstanceId，赋值给testConfig
   //设置testConfig的各个状态
   String appInstanceId = UUID.randomUUID().toString();
   testConfig.setDeployStatus(EnumTestConfigDeployStatus.DEPLOYING);
   ProjectTestConfigStageStatus stageStatus = new ProjectTestConfigStageStatus();
   testConfig.setStageStatus(stageStatus);
   testConfig.setAppInstanceId(appInstanceId);
   //配置用于访问lcm组件的token，此token从发起部署请求的request http header中取到，key为access token
   testConfig.setLcmToken(token);
   int tes = projectMapper.updateTestConfig(testConfig);
```

```
  ...
  // update project status
  ApplicationProject project = projectMapper.getProject(userId, projectId);
  project.setStatus(EnumProjectStatus.DEPLOYING);
  project.setLastTestId(testConfig.getTestId());
  int res = projectMapper.updateProject(project);
  if (res < 1) {
  LOGGER.error("Update project {} in db failed.", project.getId());
  FormatRespDto error = new FormatRespDto(Status.BAD_REQUEST, "update product in db failed.");
  return Either.left(error);
  }
  return Either.right(projectMapper.getProject(userId, projectId));
  }
```

- **执行定时任务**

实现代码位于org/edgegallery/developer/util/ScheduleTask.java

```
@Component
@Lazy(false)
public class ScheduleTask {
 //四个service代表4种需要定时任务的业务场景
 @Autowired
 private TestCaseService testCaseService;
 @Autowired
 private UploadFileService uploadFileService;
 @Autowired
 private ProjectService projectService;
 @Autowired
 private VmService vmService;
 //部署任务，每30秒执行一次
 @Scheduled(cron = "0/30 * * * * ?")
 public void processConfigDeploy() {
 projectService.processDeploy();
 }
 //...
}
```

业务逻辑为，找到所有处于DEPLOYING状态的testConfig，执行之:

```
public void processDeploy() {
 // get deploying config list from db
 List<ProjectTestConfig> configList = projectMapper
```

```
4   .getTestConfigByDeployStatus(EnumTestConfigDeployStatus.DEPLOYING.toStri
ng());
5   if (CollectionUtils.isEmpty(configList)) {
6   return;
7   }
8   configList.forEach(this::processConfig);
9   }
10  }
11
12  public void processConfig(ProjectTestConfig config) {
13   String nextStage = config.getNextStage();
14   if (StringUtils.isBlank(nextStage)) {
15   return;
16   }
17   try {
18   IConfigDeployStage stageService = deployServiceMap.get(nextStage + "_se
rvice");
19   stageService.execute(config);
20   } catch (Exception e) {
21   LOGGER.error("Deploy project config:{} failed on stage :{}, res:{}", co
nfig.getTestId(), nextStage,
22   e.getMessage());
23   }
24  }
```

可以看到其中定义了**IConfigDeployStage接口**，位于用于org.edgegallery.developer.service.deploy，IConfigDeployStage描述deploy任务

```
1  public interface IConfigDeployStage {
2   //参数为testconfig
3   boolean execute(ProjectTestConfig config) throws InterruptedException;
4   boolean destroy();
5   boolean immediateExecute(ProjectTestConfig config);
6  }
```

IConfigDeployStage接口的四个实现类代表部署的4个不同步骤，其顺序为：

1. StageCreateCsar：根据yaml配置创建应用的csar包
2. StageSelectHost：选择一个沙箱环境，set进testConfig，此环境将用于应用实例化
3. StageInstantiate：调用lcm接口向沙箱环境实例化应用
4. StageWorkStatus：调用lcm获取app部署后的workload信息，写入testConfig

并依次执行。在执行过程中，同步更新project关联的testConfig的内容，供下一阶段使用。首先执行的是**StageCreateCsar**:

```
@Service("csar_service")
public class StageCreateCsar implements IConfigDeployStage {
 //...
 //下一阶段的实例
 @Resource(name = "hostInfo_service")
 private IConfigDeployStage stageService;
 @Override
 public boolean execute(ProjectTestConfig config) throws InterruptedExcep
tion {
 //...get project info
  try {
 // create csar package, impl by csar creator
 //目录位于projectPath+appInstanceId
 projectService.createCsarPkg(userId, project, config);
 csarStatus = EnumTestConfigStatus.Success;
 processSuccess = true;
 } catch (Exception e) {
 processSuccess = false;//...
 } finally {
 //更新project与关联的testConfig中的信息，将stageStatus字段中的csar置ture
 projectService.updateDeployResult(config, project, "csar", csarStatus);
 }
 //如果成功，执行select host
 return processSuccess == true ? stageService.execute(config) : processS
uccess;
 }
 ...
}
```

后面的整体结构相同，**StageSelectHost**主要用于分配测试节点：

```
@Service("hostInfo_service")
public class StageSelectHost implements IConfigDeployStage {
 //...
 @Resource(name = "instantiateInfo_service")
 private IConfigDeployStage instantiateService;
 @Override
 public boolean execute(ProjectTestConfig config) throws InterruptedExcep
tion {
 //...get project info
```

```
//...why sleep ?
 //如果是本地环境
 if (config.isPrivateHost()) {
 List<MepHost> privateHosts = hostMapper.getHostsByUserId(project.getUserId());
 //写入testConfig的host域
 config.setHosts(privateHosts.subList(0, 1));
 hostStatus = EnumTestConfigStatus.Success;
 processSuccess = true;
 } else {
 //admin下的状态为normal的
 List<MepHost> enabledHosts = hostMapper
 .getHostsByStatus(EnumHostStatus.NORMAL, "admin", project.getPlatform().get(0), "K8S");
 if (CollectionUtils.isEmpty(enabledHosts)) {
 ...
 } else {
 processSuccess = true;
 enabledHosts.get(0).setPassword("");
 //向testConfig中注入可用的host
 config.setHosts(enabledHosts.subList(0, 1));
 hostStatus = EnumTestConfigStatus.Success;
 }
 }
 //更新project与关联的testConfig中的关于hostInfo的状态
 projectService.updateDeployResult(config, project, "hostInfo", hostStatus);
 //继续执行
 ...
 }
}
```

继续执行实例化**StageInstantiate**，即把yaml描述的应用实例化在host上：

```
@Service("instantiateInfo_service")
public class StageInstantiate implements IConfigDeployStage {
 @Autowired
 private ProjectService projectService;
 @Autowired
 private ProjectMapper projectMapper;
 @Override
 public boolean execute(ProjectTestConfig config) {
 //...get project info
```

```
...
// check mep service dependency，检查依赖的mep能力，具体为解析json后检查能力的packageId
dependencyResult = projectService.checkDependency(project);
//...
// deploy app
File csar;
try {
//在本地读取csar包
csar = new File(projectService.getProjectPath(config.getProjectId()) + config.getAppInstanceId() + ".csar");
//执行具体实例化操作
instantiateAppResult = projectService
.deployTestConfigToAppLcm(csar, project, config, userId, config.getLcmToken());
if (!instantiateAppResult) {
LOGGER.error("Failed to instantiate app which appInstanceId is : {}.", config.getAppInstanceId());
} else {
// update status when instantiate success
config.setAppInstanceId(config.getAppInstanceId());
config.setWorkLoadId(config.getAppInstanceId());
config.setDeployDate(new Date());
processSuccess = true;
instantiateStatus = EnumTestConfigStatus.Success;
}
} catch (Exception e) {
...
} finally {
projectService.updateDeployResult(config, project, "instantiateInfo", instantiateStatus);
}
return processSuccess;
}
}
```

继续看具体的实例化实现：

```
public boolean deployTestConfigToAppLcm(File csar, ApplicationProject project, ProjectTestConfig testConfig,
String userId, String token) {
Type type = new TypeToken<List<MepHost>>() { }.getType();
//hosts即从testConfig中获取在StageSelectHost步中写入的边缘节点 host ip
```

```
  List<MepHost> hosts = gson.fromJson(gson.toJson(testConfig.getHosts()), type);
  MepHost host = hosts.get(0);
  // Note(ch) only ip?
  testConfig.setAccessUrl(host.getLcmIp());
  // upload pkg
   //调用applcm POST /lcmcontroller/v1/tenants/tenantId/packages
   //将project的csar文件等project信息上传给applcm
   LcmLog lcmLog = new LcmLog();
   String uploadRes = HttpClientUtil
   .uploadPkg(host.getProtocol(), host.getLcmIp(), host.getPort(), csar.getPath(), userId, token, lcmLog);
   //err handler ...
   Gson gson = new Gson();
   Type typeEvents = new TypeToken<UploadResponse>() { }.getType();
   //解析resp，获取applcm的pkgId
   UploadResponse uploadResponse = gson.fromJson(uploadRes, typeEvents);
   String pkgId = uploadResponse.getPackageId();
   //回填testConfig的包id
   testConfig.setPackageId(pkgId);
   projectMapper.updateTestConfig(testConfig);
   // distribute pkg
   // 调用applcm POST /lcmcontroller/v1/tenants/tenantId/packages/packageIc
   // 将package分发到host节点上去
   boolean distributeRes = HttpClientUtil
   .distributePkg(host.getProtocol(), host.getLcmIp(), host.getPort(), userId, token, pkgId, host.getMecHost(),
   lcmLog);
   //err handler...
   //获取appInstanceId后，调实例化
   String appInstanceId = testConfig.getAppInstanceId();
   // instantiate application
   boolean instantRes = HttpClientUtil
   .instantiateApplication(host.getProtocol(), host.getLcmIp(), host.getPort(), appInstanceId, userId, token,
   lcmLog, pkgId, host.getMecHost());
   ...
   return true;
   }
```

继续看app的实例化：

```
/**
```

```
2   * instantiateApplication.
3   *
4   * @return InstantiateAppResult
5   */
6   public static boolean instantiateApplication(String basePath, String app
InstanceId, String userId, String token,
7   LcmLog lcmLog, String pkgId, String mecHost, Map<String, String> inputPa
rams) {
8   //before instantiate ,call distribute result interface
9   //调用lcm GET /lcmcontroller/v1/tenants/tenantId/packages/packageId
10   //获取上一步包分发的结果
11   String disRes = getDistributeRes(basePath, userId, token, pkgId);
12   ...
13   //parse dis res
14   Gson gson = new Gson();
15   Type typeEvents = new TypeToken<List<DistributeResponse>>() {
}.getType();
16   List<DistributeResponse> list = gson.fromJson(disRes, typeEvents);
17   String appName = list.get(0).getAppPkgName();
18   //set instantiate headers
19   HttpHeaders headers = new HttpHeaders();
20   headers.setContentType(MediaType.APPLICATION_JSON);
21   headers.set(Consts.ACCESS_TOKEN_STR, token);
22   //set instantiate bodys
23   //创建init 应用所需的body，调用 lcm POST /lcmcontroller/v1/tenants/tenant
Id/app_instances/appInstanceId/instantiate
24   InstantRequest ins = new InstantRequest();
25   ins.setAppName(appName);
26   ins.setHostIp(mecHost);
27   ins.setPackageId(pkgId);
28   ins.setParameters(inputParams);
29   LOGGER.warn(gson.toJson(ins));
30   HttpEntity<String> requestEntity = new HttpEntity<>(gson.toJson(ins), h
eaders);
31   String url = basePath +
Consts.APP_LCM_INSTANTIATE_APP_URL.replaceAll("appInstanceId", appInstanceI
d)
32   .replaceAll("tenantId", userId);
33   LOGGER.warn(url);
34   ResponseEntity<String> response;
35   try {
36   REST_TEMPLATE.setErrorHandler(new CustomResponseErrorHandler());
```

```
37  response = REST_TEMPLATE.exchange(url, HttpMethod.POST, requestEntity,
String.class);
38  LOGGER.info("APPlCM instantiate log:{}", response);
39  } ...
40  if (response.getStatusCode() == HttpStatus.OK) {
41  return true;
42  }
43  LOGGER.error("Failed to instantiate application which appInstanceId is
{}", appInstanceId);
44  return false;
45  }
```

最后，执行**StageWorkStatus**，获取app部署后的实例信息，反写testConfig：

```
1  @Override
2  public boolean execute(ProjectTestConfig config) throws InterruptedExcep
tion {
3  boolean processStatus = false;
4  EnumTestConfigStatus status = EnumTestConfigStatus.Failed;
5  //
6  ApplicationProject project = projectMapper.getProjectById(config.getProj
ectId());
7  String userId = project.getUserId();
8  Type type = new TypeToken<List<MepHost>>() { }.getType();
9  List<MepHost> hosts = gson.fromJson(gson.toJson(config.getHosts()),
type);
10  MepHost host = hosts.get(0);
11  //...sleep 10000 ms
12  //调用lcm GET /lcmcontroller/v1/tenants/tenantId/app_instances/appInstar
ceId
13  //获取app信息
14  String workStatus = HttpClientUtil
15  .getWorkloadStatus(host.getProtocol(), host.getLcmIp(), host.getPort(),
config.getAppInstanceId(), userId,
16  config.getLcmToken());
17  LOGGER.info("pod workStatus: {}", workStatus);
18  //调用lcm GET /lcmcontroller/v1/tenants/tenantId/app_instances/appInstar
ceId/workload/events
19  //获取app部署后的workload事件
20  String workEvents = HttpClientUtil
21  .getWorkloadEvents(host.getProtocol(), host.getLcmIp(), host.getPort(),
config.getAppInstanceId(), userId,
22  config.getLcmToken());
23  LOGGER.info("pod workEvents: {}", workEvents);
```

```
24  if (workStatus == null || workEvents == null) {
25  // compare time between now and deployDate
26  long time = System.currentTimeMillis() -
config.getDeployDate().getTime();
27  LOGGER.info("over time:{}, wait max time:{}, start time:{}", time, MAX_
SECONDS,
28  config.getDeployDate().getTime());
29  if (config.getDeployDate() == null || time > MAX_SECONDS * 1000) {
30  config.setErrorLog("Failed to get workloadStatus: pull images failed
");
31  String message = "Failed to get workloadStatus after wait {} seconds wh
ich appInstanceId is : {}";
32  LOGGER.error(message, MAX_SECONDS, config.getAppInstanceId());
33  } else {
34  return true;
35  }
36  } else {
37  processStatus = true;
38  status = EnumTestConfigStatus.Success;
39  //merge workStatus and workEvents
40  //获取app的部署后的所有pod信息
41  String pods = mergeStatusAndEvents(workStatus, workEvents);
42  config.setPods(pods);
43  //set access url
44  //根据workStatus解析出pod的service配置，返回lcmIp+service.NodePort信息，更
新testConfig
45  String accsessUrl = getAccessUrl(host, workStatus);
46  if (accsessUrl != null) {
47  config.setAccessUrl(accsessUrl.substring(0, accsessUrl.length() - 1));
48  }
49  LOGGER.info("Query workload status response: {}", workStatus);
50  }
51  // update test-config
52  projectService.updateDeployResult(config, project, "workStatus",
status);
53  return processStatus;
54  }
```

应用发布

应用发布主要涉及以下三步，分别为

1. 应用配置 上传说明文档，为应用配置mp2相关接口以及在服务发布配置中定义对外的api信息
2. 应用认证 调用atp模块的test case 相关接口，执行test case任务
3. 应用发布

1. 应用配置

发布配置的接口包括

- 10. ReleaseConfig
  - 10.1 GET release config
  - 10.2 POST release config
  - 10.3 PUT release config

其中对于csar包的解析接口位于：

- 9. AppRelease
  - 9.1 GET pkg structure
  - 9.2 GET file content

```
//发布配置
public class ReleaseConfig {
 private String releaseId;
 private String projectId;
 private String guideFileId;
 private String appInstanceId;
 private CapabilitiesDetail capabilitiesDetail;
 private AtpResultInfo atpTest;
 private String testStatus;
  private Date createTime;
  ...
}
```

其中创建release config的实现基本为db操作，同时重写的csar包：

```
 /**
 * saveConfig.
 */
 public Either<FormatRespDto, ReleaseConfig> saveConfig(String projectId, ReleaseConfig config) {
 ...// project id check
 ...// release config check ,if exists, return
 //创建releaseConfig
 String releaseId = UUID.randomUUID().toString();
 config.setReleaseId(releaseId);
  config.setProjectId(projectId);
```

```
11  config.setCreateTime(new Date());
12  int res = configMapper.saveConfig(config);
13  ...
14  ApplicationProject applicationProject = projectMapper.getProjectById(pr
ojectId);
15  //如果app中集成了mep能力，重构csar包
16  if (!CollectionUtils.isEmpty(applicationProject.getCapabilityList()) ||
!CapabilitiesDetail
17  .isEmpty(config.getCapabilitiesDetail()) ||
!StringUtils.isEmpty(config.getGuideFileId())) {
18  if (applicationProject.getDeployPlatform() == EnumDeployPlatform.KUBERN
ETES) {
19  //重构k8s csar包，加入说明文档 appd等依赖关系
20  Either<FormatRespDto, Boolean> rebuildRes = rebuildCsar(projectId, conf
ig);
21  if (rebuildRes.isLeft()) {
22  return Either.left(rebuildRes.getLeft());
23  }
24  } else {
25  //重构虚机 csar包
26  Either<FormatRespDto, Boolean> rebuildRes = rebuildVmCsar(projectId, co
nfig);
27  if (rebuildRes.isLeft()) {
28  return Either.left(rebuildRes.getLeft());
29  }
30  }
31  }
32  ...
33  }
```

2. 应用认证

执行测试规范，如果选择第三方测试，则从atp平台拉取相应的test case，接口涉及:

3.6 GET query all test cases under one scneario

edgegallery/atp/v1/testscenarios/testcases 查找testcsase

之后根据选择的tset case，调用atp平台接口，执行测试用例

1.2 POST run test task /edgegallery/atp/v1/tasks/{taskId}/action/run

以上接口实现将在atp部分详解，此处略过

3. 应用发布

具体发布时，分为两个场景(待验证):

- 发布project到应用市场 3.11 POST upload to store

/mec/developer/v1/projects/{projectId}/action/upload

- 将project发布为mep平台服务 3.16 POST open project api

首先看发布到appStore的操作：

```
/**
 * uploadToAppStore.
 *
 * @return
 */
public Either<FormatRespDto, Boolean> uploadToAppStore(String userId, String projectId, String userName,
String token) {
// 0 check data. must be tested, and deployed status must be ok, can not be error.
ApplicationProject project = projectMapper.getProject(userId, projectId);
// err check ...
ReleaseConfig releaseConfig = configMapper.getConfigByProjectId(projectId);
//...test case status check
//调 appStore POST /mec/appstore/v1/apps?userId={userId}&userName={userName}，上传csar文件
Either<FormatRespDto, JsonObject> resCsar = getCsarAndUpload(projectId, project, releaseConfig, userId,
userName, token);
...
JsonObject jsonObject = resCsar.getRight();
...
//调appStore POST /mec/appstore/v1/apps/{appId}/packages/{packageId}/action/publish, 其中appId来自jsonObject
Either<FormatRespDto, Boolean> pubRes = publishApp(jsonObject, token);...
//获取依赖的mep平台服务
CapabilitiesDetail capabilitiesDetail = releaseConfig.getCapabilitiesDetail();
if (capabilitiesDetail.getServiceDetails() != null && !capabilitiesDetail.getServiceDetails().isEmpty()) {
//save db to openmepcapabilitydetail
//open mep capability 即mep 平台服务，在db里记录app的依赖
List<String> openCapabilityIds = new ArrayList<>();
for (ServiceDetail serviceDetail : capabilitiesDetail.getServiceDetails()) {
OpenMepCapabilityDetail detail = new OpenMepCapabilityDetail();
//new mep cap group
OpenMepCapabilityGroup group = new OpenMepCapabilityGroup();
```

```
31  String groupId = UUID.randomUUID().toString();
32  //填充信息
33  fillCapabilityGroup(serviceDetail, groupId, group);
34  fillCapabilityDetail(serviceDetail, detail, jsonObject, userId,
groupId);
35  //db中保存 mep cap能力信息
36  Either<FormatRespDto, Boolean> resDb = doSomeDbOperation(group, detail,
serviceDetail,
37  openCapabilityIds);
38  if (resDb.isLeft()) {
39  return Either.left(resDb.getLeft());
40  }
41  }
42  project.setOpenCapabilityId(openCapabilityIds.toString());
43  project.setStatus(EnumProjectStatus.RELEASED);
44  int updRes = projectMapper.updateProject(project);
45  ...
46  }
47  //更新服务状态
48  project.setStatus(EnumProjectStatus.RELEASED);
49  int updRes = projectMapper.updateProject(project);
50  ...
51  return Either.right(true);
52  }
```

发布为mep平台服务的实现，主要是一些写表操作，不再赘述：

```
1 /**
2  * openToMecEco.
3  *
4  * @return
5  */
6  public Either<FormatRespDto, OpenMepCapabilityGroup> openToMecEco(String
userId, String projectId) {
7  ApplicationProject project = projectMapper.getProject(userId,
projectId);
8  // verify app project and test config
9  ...
10  // if has opened, delete before
11  //如果project中依赖了mep服务，先删除
12  String openCapabilityDetailId = project.getOpenCapabilityId();
13  if (openCapabilityDetailId != null) {
14  openMepCapabilityMapper.deleteCapability(openCapabilityDetailId);
```

```
15  }
16  //组合信息
17  OpenMepCapabilityGroup capabilityGroup = openMepCapabilityMapper.getEco
GroupByName(project.getType());
18  String groupId;
19  //如果没有该类型，创建之
20  if (capabilityGroup == null) {
21  OpenMepCapabilityGroup group = new OpenMepCapabilityGroup();
22  groupId = UUID.randomUUID().toString();
23  group.setGroupId(groupId);
24  group.setOneLevelName(project.getType());
25  group.setType(EnumOpenMepType.OPENMEP_ECO);
26  group.setDescription("Open MEP ecology group.");
27
28  int groupRes = openMepCapabilityMapper.saveGroup(group);
29  if (groupRes < 1) {
30  LOGGER.error("Create capability group failed {}", group.getGroupId());
31  FormatRespDto error = new FormatRespDto(Status.BAD_REQUEST, "create cap
ability group failed");
32  return Either.left(error);
33  }
34  } else {
35  groupId = capabilityGroup.getGroupId();
36  }
37
38  OpenMepCapabilityDetail detail = new OpenMepCapabilityDetail();
39  detail.setDetailId(UUID.randomUUID().toString());
40  detail.setGroupId(groupId);
41  detail.setService(project.getName());
42  detail.setVersion(project.getVersion());
43  detail.setDescription(project.getDescription());
44  detail.setProvider(project.getProvider());
45  detail.setApiFileId(test.getAppApiFileId());
46  SimpleDateFormat time = new SimpleDateFormat("yyyy-MM-dd HH:mm");
47  detail.setUploadTime(time.format(new Date()));
48
49  int detailRes = openMepCapabilityMapper.saveCapability(detail);
50  ... //err handler
51  OpenMepCapabilityGroup result = openMepCapabilityMapper.getOpenMepCapab
ilitiesByGroupId(groupId);
52  ... //err handler
```

```
53  project.setOpenCapabilityId(detail.getDetailId());
54  int updateRes = projectMapper.updateProject(project);
55  ... //err handler
56  LOGGER.info("Open {} to Mec Success", groupId);
57  return Either.right(result);
58  }
```

沙箱管理

沙箱管理用于关联一个沙箱集群，在部署调测时，将应用部署到此沙箱后调测。涉及的接口有：

以创建为例：

```
1  @Transactional
2  public Either<FormatRespDto, Boolean> createHost(MepCreateHost host, Str
ing token) {
3  ...//param check
4  ...
5  //health check
6  String healRes = HttpClientUtil.getHealth(host.getProtocol(), host.getLc
mIp(), host.getPort());
7  ...
8  // add mechost to lcm
9  //调用 lcm POST /lcmcontroller/v1/hosts 接口
10  boolean addMecHostRes = addMecHostToLcm(host);
11  ...
12  // 如果表单上有上传文件接口，上传之后返回一个fileId，作为configId
13  // 调用 POST /lcmcontroller/v1/configuration接口向lcm发送config文件
14  if (StringUtils.isNotBlank(host.getConfigId())) {
15  // upload file
16  UploadedFile uploadedFile = uploadedFileMapper.getFileById(host.getConf
igId());
17  boolean uploadRes = uploadFileToLcm(host.getLcmIp(), host.getPort(), up
loadedFile.getFilePath(),
18  host.getMecHost(), token);
19  //...
```

```
20  }
21  //db 操作
22  host.setHostId(UUID.randomUUID().toString()); // no need to set hostId
by user
23  host.setVncPort(VNC_PORT);
24  int ret = hostMapper.createHost(host);
25  ...
26  }
```

其他

问：appInstance和project的关系?

答：project表示一个app的定义内容，为静态，类似class，appInstance表示根据project的定义，进行的一次实例化。

问：mep平台服务中，api通过file描述，相应的fileId何时生成？何时关联?

答：服务间的http调用，如何保证全局的数据一致性

以下api的调用时机

- 2. App
  - 2.1 POST upload app
  - 2.2 GET all test task
  - 2.3 GET all app tags
  - 2.4 POST upload app to store
  - 2.5 GET start test
  - 2.6 GET subtask list
- 7. LocalApi
  - 7.1 GET one api file
- 8. Health
  - 8.1 GET health
- 11. DeployConfig
  - 11.1 GET deploy platform
  - 11.2 PUT deploy platform